

US 6,981,047 B2

1

METHOD AND APPARATUS FOR PROVIDING MOBILE AND OTHER INTERMITTENT CONNECTIVITY IN A COMPUTING ENVIRONMENT

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a division of application Ser. No. 09/330,310, filed Jun. 11, 1999 now U.S. Pat. No. 6,546,425, entitled "Method And Apparatus For Providing Mobile And Other Intermittent Connectivity In A Computing Environment" which claims the benefit of provisional application No. 60/103,598 filed Oct. 9, 1998 entitled "Method and Apparatus For Providing Wireless Connectivity In A Computing Environment" the entire content of each of which is hereby incorporated by reference in this application.

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

Not applicable

FIELD OF THE INVENTION

The present invention relates to connectivity between networked computing devices. More particularly, the present invention relates to methods and systems that transparently address the characteristics of nomadic systems, and enable existing network applications to run reliably in the associated mobile environments. Still more particularly, the invention relates to techniques and systems for providing a continuous data stream connection between intermittently-connected devices such as handheld data units and personal computing devices.

BACKGROUND AND SUMMARY OF THE INVENTION

Increasingly, companies are seeing rapid access to key information as the way to maintaining a competitive advantage. To provide immediate access to this information, mobile and other intermittently-connected computing devices are quickly and swiftly becoming an essential part of corporate networks—especially with the proliferation of inexpensive laptops and hand-held computing devices. However, integrating these nomadic devices into existing network infrastructures has created a challenge for the information manager.

Many problems in mobile networking parallel the difficulties in early local area networks (LANs) before the adoption of Ethernet. There are a variety of mobile protocols and interfaces, and because standards are just developing, there is little interoperability between systems. In addition, performance over these network technologies has been typically slow and bandwidth limited. Implementation costs to date have been high due to the specialized nature of deployed systems.

Along with these issues, mobile technologies present a category of problems unto their own. Interconnects back into the main network may travel over and through a public network infrastructure, thus allowing sensitive information to possibly be tapped into. Furthermore, if any of the intermediary interconnects are via a wireless interface, the information is actually broadcast, and anyone with a similar interface can eavesdrop without much difficulty.

But, perhaps even more significantly, mobile networking has generally in the past been limited to mostly message-oriented or stateless applications—and thus has not been

2

readily adaptable for existing or new corporate applications that use client/server, host-terminal, web-based or shared file systems models. This is because such commonly used applications need stateful sessions that employ a continuous stream of data—not just a stateless packet exchange—to work effectively and reliably.

To this end, many or most popular off-the-shelf networking applications require TCP/IP sessions, or private virtual circuits. These sessions cannot continue to function if they encounter network interruptions, nor can they tolerate roaming between networks (i.e., a change of network addresses) while established. Yet, mobile networking is, by its nature, dynamic and unreliable. Consider these common scenarios encountered in mobile networks:

Disconnected or Out of Range User

When a mobile device disconnects from a given network or loses contact (e.g., through an outage or "hole" in the coverage of a wireless interconnect), the session-oriented application running on the mobile device loses its stateful connection with its peer and ceases to operate. When the device is reattached or moves back into contact, the user must re-connect, log in again for security purposes, find the place in the application where work was left off, and possibly re-enter lost data. This reconnection process is time consuming, costly, and can be very frustrating.

Moving to a Different Network or Across a Router Boundary (Network Address Change)

Mobile networks are generally segmented for manageability purposes. But the intent of mobile devices is to allow them to roam. Roaming from one network interconnect to another can mean a change of network address. If this happens while the system is operational, the routing information must be changed for communications to continue between the associated peers. Furthermore, acquiring a new network address may require all of the previously established stateful application sessions to be terminated—again presenting the reconnection problems noted above.

Security

As mentioned before, companies need to protect critical corporate data. Off-the-shelf enterprise applications are often written with the assumption that access to the physical network is controlled (i.e., carried within cables installed inside a secure facility), and security is maintained through an additional layer of authentication and possible encryption. These assumptions have not been true in the nomadic computing world—where data is at risk for interception as it travels over public airways or public wire-line infrastructures.

Summary

It would be highly desirable to provide an integrated solution that transparently addresses the characteristics of nomadic systems, and enables existing network applications to run reliably in these mobile environments.

A presently preferred embodiment of the present invention solves this problem by providing a seamless solution that extends the enterprise network, letting network managers provide mobile users with easy access to the same applications as stationary users without sacrificing reliability or centralized management. The solution combines advantages of present-day wire-line network standards with emerging mobile standards to create a solution that works with existing network applications.

In accordance with one aspect of a presently preferred embodiment of the present invention, a Mobility Management Server (MMS) coupled to the mobile interconnect maintains the state of each of any number of Mobile End Systems (MES) and handles the complex session manage-

US 6,981,047 B2

3

ment required to maintain persistent connections to the network and to peer application processes. If a Mobile End System becomes unreachable, suspends, or changes network address (e.g., due to roaming from one network interconnect to another), the Mobility Management Server maintains the connection to the associated peer—allowing the Mobile End System to maintain a continuous virtual connection even though it may temporarily lose its actual physical connection.

A presently preferred exemplary embodiment of the present invention also provides the following (among others) new and advantageous techniques and arrangements:

- a Mobility Management Server, providing user configurable session priorities for mobile clients;
- per-user mobile policy management for managing consumption of network resources,
- a roaming methodology making use of the industry standard Dynamic Host Configuration Protocol (DHCP) in coordination with a Mobility Management Server;
- automatic system removal of unreliable datagrams based on user configurable timeouts; and
- automatic system removal of unreliable datagrams based on user configurable retries

In more detail, a presently preferred exemplary embodiment of the present invention in one of its aspects provides a Mobility Management Server that is coupled to the mobile interconnect (network). The Mobility Management Server maintains the state of each of any number of Mobile End Systems and handles the complex session management required to maintain persistent connections to the network and to other processes (e.g., running on other network-based peer systems). If a Mobile End System becomes unreachable, suspends, or changes network address (e.g., due to roaming from one network interconnect to another), the Mobility Management Server maintains the connection to the associated peer, by acknowledging receipt of data and queuing requests. This proxying by the Mobility Management Server allows the application on the Mobile End System to maintain a continuous connection even though it may temporarily lose its physical connection to a specific network medium.

In accordance with another aspect of a presently preferred exemplary embodiment of the present invention, a Mobility Management Server manages addresses for Mobile End Systems. Each Mobile End System is provided with a proxy address on the primary network. This highly available address is known as the "virtual address" of the Mobile End System. The Mobility Management Server maps the virtual addresses to current "point of presence" addresses of the nomadic systems. While the point of presence address of a Mobile End System may change when the mobile system changes from one network interconnect to another, the virtual address stays constant while any connections are active or longer if the address is statically assigned.

In accordance with yet another aspect of a presently preferred exemplary embodiment of the present invention, a Mobility Management Server provides centralized system management of Mobile End Systems through a console application and exhaustive metrics. A presently preferred exemplary embodiment of the present invention also provides user configurable session priorities for mobile clients running through a proxy server, and per-user mobile policy management for managing consumption of network resources.

In accordance with yet another aspect of a presently preferred exemplary embodiment of the present invention, a

4

Remote Procedure Call protocol and an Internet Mobility Protocol are used to establish communications between the proxy server and each Mobile End System.

Remote procedure calls provide a method for allowing a process on a local system to invoke a procedure on a remote system. The use of the RPC protocol allows Mobile End Systems to disconnect, go out of range or suspend operation without losing active network sessions. Since session maintenance does not depend on a customized application, off-the-shelf applications will run without modification in the nomadic environment.

The Remote Procedure Call protocol generates transactions into messages that can be sent via the standard network transport protocol and infrastructure. These RPC messages contain the entire network transaction initiated by an application running on the Mobile End System—enabling the Mobility Management Server and Mobile End System to keep connection state information synchronized at all times—even during interruptions of the physical link connecting the two. In the preferred embodiment of a presently preferred exemplary embodiment of the present invention providing RPC's, the proxy server and the Mobile End Systems share sufficient knowledge of each transaction's state to maintain coherent logical database about all shared connections at all times.

The Internet Mobility Protocol provided in accordance with a presently preferred exemplary embodiment of the present invention compensates for differences between wired local area network interconnects and other less reliable networks such as a wireless LAN or WAN. Adjusted frame sizes and protocol timing provide significant performance improvements over non-mobile-aware transports—dramatically reducing network traffic. This is important when bandwidth is limited or when battery life is a concern.

The Internet Mobility Protocol provided in accordance with a presently preferred exemplary embodiment of the present invention also ensures the security of organizational data as it passes between the Mobile End System and the Mobility Management Server over public network interconnects or airways. The Internet Mobility Protocol provides a basic firewall function by allowing only authenticated devices access to the organizational network. The Internet Mobility Protocol can also certify and encrypt all communications between the Mobility Management Server and the Mobile End System.

In accordance with yet another aspect of a presently preferred exemplary embodiment of the present invention, mobile inter-connectivity is built on standard transport protocols (e.g., TCP/IP, UDP/IP and DCCP, etc.) to extend the reach of standard network application interfaces. A presently preferred exemplary embodiment of the present invention efficiently integrates transport, security, address management, device management and user management needs to make nomadic computing environments effectively transparent. The Internet Mobility Protocol provides an efficient mechanism for multiplexing multiple streams of data (reliable and unreliable) through a single virtual channel provided by such standard transport protocols over standard network infrastructure.

With the help of the RPC layer, the Internet Mobility Protocol coalesces data from different sources targeted for the same or different destinations, together into a single stream and forwards it over a mobile link. At the other end of the mobile link, the data is demultiplexed back into multiple distinct streams, which are sent on to their ultimate destination(s). The multiplexing/demultiplexing technique allows for maximum use of available bandwidth (by gener-

US 6,981,047 B2

5

ating the maximum sized network frames possible), and allows multiple channels to be established (thus allowing prioritization and possibly providing a guaranteed quality of service if the underlying network provides the service)

The Internet Mobility Protocol provided in accordance with a presently preferred exemplary embodiment of the present invention provides the additional features and advantages, for example:

Transport protocol independence

Allows the network point of presence (POP) or network infrastructure to change without affecting the flow of data (except where physical boundary, policy or limitations of bandwidth may apply)

Minimal additional overhead

Automatic fragment resizing to accommodate the transmission medium. (When the protocol data unit for a given frame is greater than the available maximum transmission unit of the network medium, the Internet Mobility Protocol will fragment and reassemble the frame to insure that it can traverse the network. In the event of a retransmit, the frame will again be assessed. If the network infrastructure or environment changes, the frame will be refragmented or in the case that the maximum transmission unit actually grew, sent as a single frame.)

Semantics of unreliable data are preserved, by allowing frames to discard unreliable data during retransmit

Provides a new semantic of Reliable Datagram service (Delivery of datagrams can now be guaranteed to the peer terminus of the Internet Mobility Protocol connection. Notification of delivery can be provided to a requesting entity)

Considers the send and receive transmission path separately, and automatically tailors its operating parameters to provided optimum throughput. (Based on hysteresis, it adjusts such parameters as frame size/fragmentation threshold, number of frames outstanding (window), retransmit time, and delayed acknowledgement time to reduce the amount of duplicate data sent through the network.)

Network fault tolerant (since the expected usage is in a mobile environment, temporary loss of network medium connectivity does not result in a termination of the virtual channel or application based connection)

Provides an in-band signaling method to its peer to adjust operating parameters (each end of the connection can alert its peer to any changes in network topology or environment)

Employs congestion avoidance algorithms and gracefully decays throughput when necessary

Employs selective acknowledgement and fast retransmit policies to limit the number of gratuitous retransmissions, and provide faster handoff recovery in nomadic environments (This also allows the protocol to maintain optimum throughput in a lossy network environment)

Employs sliding window technology to allow multiple frames to be outstanding (This parameter is adjustable in each direction and provides for streaming frames up to a specified limit without requiring an acknowledgement from its peer)

Sequence numbers are not byte oriented, thus allowing for a single sequence number to represent up to a maximum payload size

Security aware (Allows for authentication layer and encryption layer to be added in at the Internet Mobility Protocol layer)

6

Compression to allow for better efficiency through bandwidth limited links

Balanced design, allowing either peer to migrate to a new point of presence

Either side may establish a connection to the peer

Allows for inactivity timeouts to be invoked to readily discard dormant connections and recover expended resources

Allows for a maximum lifetime of a given connection (e.g., to allow termination and/or refusal to accept connections after a given period or time of day)

A presently preferred exemplary embodiment of the present invention also allows a system administrator to manage consumption of network resources. For example, the system administrator can place controls on Mobile End Systems, the Mobility Management Server, or both. Such controls can be for the purpose, for example, of managing allocation of network bandwidth or other resources, or they may be related to security issues. It may be most efficient to perform management tasks at the client side for clients with lots of resources. However, thin clients don't have many resources to spare, so it may not be practical to burden them with additional code and processes for performing policy management. Accordingly, it may be most practical to perform or share such policy management functions for thin clients at a centralized point such as the Mobility Management Server. Since the Mobility Management Server proxies the distinct data streams of the Mobile End Systems, it provides a central point from which to conduct policy management. Moreover, the Mobility Management Server provides the opportunity to perform policy management of Mobile End Systems on a per user and/or per device basis. Since the Mobility Management Server is proxying on a per user basis, it has the ability to control and limit each user's access to network resources on a per-user basis as well as on a per-device basis.

As one simple example, the Mobility Management Server can "lock out" certain users from accessing certain network resources. This is especially important considering that interface network is via a mobile interconnect, and may thus "extend" outside of the boundaries of a locked organizational facility (consider, for example, an ex-employee who tries to access the network from outside his former employer's building). However, the policy management provided by the Mobility Management Server can be much more sophisticated. For example, it is possible for the Mobility Management Server to control particular Web URL's particular users can visit, filter data returned by network services requests, and/or compress data for network bandwidth conservation. This provides a way to enhance existing and new application-level services in a seamless and transparent manner.

A presently preferred exemplary embodiment of the present invention thus extends the enterprise network, letting network managers provide mobile users with easy access to the same applications as stationary users without sacrificing reliability or centralized management. The solution combines advantages of existing wire-line network standards with emerging mobility standards to create a solution that works with existing network applications.

BRIEF DESCRIPTION OF THE DRAWINGS

These, as well as other features and advantages of this invention, will be more completely understood and appreciated by careful study of the following more detailed description of presently preferred example embodiments of

US 6,981,047 B2

7

the invention taken in conjunction with the accompanying drawings, of which:

FIG 1 is a diagram of an overall mobile computing network provided in accordance with a presently preferred exemplary embodiment of the present invention;

FIG 2 shows an example software architecture for a Mobile End System and a Mobility Management Server;

FIG 2A shows example steps performed to transfer information between a Mobile End System and a Mobility Management Server;

FIG 3 shows an example mobile interceptor architecture;

FIG 3A is a flowchart of example steps performed by the mobile interceptor;

FIG 3B is a flowchart of example steps performed by an RPC engine to handle RPC work requests;

FIGS 4-5C are flowcharts of example steps to process RPC work requests;

FIG 6 is a diagram of an example received work request;

FIG 7 is a diagram showing how a received work request can be dispatched onto different priority queues;

FIGS 8 and 9 show processing of the contents of the different priority queues;

FIGS. 10A-15B show example steps performed to provide an Internet Mobility Protocol;

FIG 16 shows example listener data structures; and

FIGS 17, 17A and 18 are flowcharts of example steps performed to provide for mobile interconnect roaming

DETAILED DESCRIPTION OF PRESENTLY PREFERRED EXAMPLE EMBODIMENTS

FIG 1 is an example of mobile enhanced networked computer system 100 provided in accordance with a presently preferred exemplary embodiment of the present invention. Networked computer system 100 includes a Mobility Management Server 102 and one or more Mobile End Systems 104. Mobile End Systems 104 can communicate with Mobility Management Server 102 via a local area network (LAN) 108. Mobility Management Server 102 serves as network level proxy for Mobile End Systems 104 by maintaining the state of each Mobile End System, and by handling the complex session management required to maintain persistent connections to any peer systems 110 that host network applications—despite the interconnect between Mobile End Systems 104 and Mobility Management Server 102 being intermittent and unreliable. In the preferred embodiment, Mobility Management Server 102 communicates with Mobile End Systems 104 using Remote Procedure Call and Internet Mobility Protocols in accordance with a presently preferred exemplary embodiment of the present invention.

In this particular example, Mobile End Systems 104 are sometimes but not always actively connected to Mobility Management Server 102. For example:

Some Mobile End Systems 104a-104k may communicate with Mobility Management Server 102 via a mobile interconnect (wirelessly in this case), e.g., conventional electromagnetic (e.g., radio frequency) transceivers 106 coupled to wireless (or wire-line) local area or wide area network 108. Such mobile interconnect may allow Mobile End Systems 104a-104k to "roam" from one cover area 107a to another coverage area 107k. Typically, there is a temporary loss of communications when a Mobile End System 104 roams from one coverage area 107 to another, moves out of range of the

8

closest transceiver 106, or has its signal temporarily obstructed (e.g., when temporarily moved behind a building column or the like).

Other Mobile End Systems 104l, 104m, . . . may communicate with Mobility Management Server 102 via non-permanent wire-based interconnects 109 such as docking ports, network cable connectors, or the like. There may be a temporary loss of communications when Mobile End Systems 104 are temporarily disconnected from LAN 108 by breaking connection 109, powering off the Mobile End Systems, etc.

Still other Mobile End Systems (e.g., 104n) may be nomadically coupled to Mobility Management Server 102 via a further network topography 111 such as a wide area network, a dial-up network, a satellite network, or the Internet, to name a few examples. In one example, network 111 may provide intermittent service. In another example, Mobile End Systems 104 may move from one type of connection to another (e.g., from being connected to Mobility Management Server 102 via wire-based interconnect 109 to being connected via network 111, or vice versa)—its connection being temporarily broken during the time it is being moved from one connection to another.

Mobile End Systems 104 may be standard mobile devices and off the shelf computers. For example, Mobile End System 104 may comprise a laptop computer equipped with a conventional radio transceiver and/or network cards available from a number of manufacturers. Mobile End Systems 104 may run standard network applications and a standard operating system, and communicate on the transport layer using a conventionally available suite of transport level protocols (e.g., TCP/IP suite). In accordance with the present invention, Mobile End Systems 104 also execute client software that enables them to communicate with Mobility Management Server 102 using Remote Procedure Call and Internet Mobility Protocols that are transported using the same such standard transport level protocols.

Mobility Management Server 102 may comprise software hosted by a conventional Windows NT or other server. In the preferred embodiment, Mobility Management Server 102 is a standards-compliant, client-server based intelligent server that transparently extends the enterprise network 108 to a nomadic environment. Mobility Management Server 102 serves as network level proxy for each of any number of Mobile End Systems 104 by maintaining the state of each Mobile End System, and by handling the complex session management required to maintain persistent connections to any peer systems 110 that host network applications—despite the mobile interconnect between Mobile End Systems 104 and transceivers 106 being intermittent and unreliable.

For example, server 102 allows any conventional (e.g., TCP/IP based) network application to operate without modification over mobile connection. Server 102 maintains the sessions of Mobile End Systems 104 that disconnect, go out of range or suspend operation, and resumes the sessions when the Mobile End System returns to service. When a Mobile End System 104 becomes unreachable, shuts down or changes its point of presence address, the Mobility Management Server 102 maintains the connection to the peer system 110 by acknowledging receipt of data and queuing requests until the Mobile End System once again becomes available and reachable.

Server 102 also extends the management capabilities of wired networks to mobile connections. Each network software layer operates independently of others, so the solution can be customized to the environment where it is deployed.

US 6,981,047 B2

9

As one example, Mobility Management Server 102 may be attached to a conventional organizational network 108 such as a local area network or wide area network. Network 108 may be connected to a variety of fixed-end systems 110 (e.g., one or most host computers 110). Mobility Management Server 102 enables Mobile End Systems 104 to communicate with Fixed End System(s) 110 using continuous session type data streams even though Mobile End Systems 104 sometimes lose contact with their associated network interconnect or move from one network interconnect 106, 109, 111 to another (e.g., in the case of wireless interconnect, by roaming from one wireless transceiver 106 coverage area 107 to another).

A Mobile End System 104 establishes an association with the Mobility Management Server 102, either at startup or when the Mobile End System requires network services. Once this association is established, the Mobile End System 104 can start one or more network application sessions, either serially or concurrently. The Mobile End System 104-to-Mobility Management Server 102 association allows the Mobile End System to maintain application sessions when the Mobile End System, disconnects, goes out of range or suspends operation, and resume sessions when the Mobile End System returns to service. In the preferred embodiment, this process is entirely automatic and does not require any intervention on the user's part.

In accordance with an aspect of a presently preferred exemplary embodiment of the present invention, Mobile End Systems 104 communicate with Mobility Management Server 102 using conventional transport protocols such as, for example, UDP/IP. Use of conventional transport protocols allows Mobile End Systems 104 to communicate with Mobility Management Server 102 using the conventional routers 112 and other infrastructure already existing on organization's network 108. In accordance with a presently preferred exemplary embodiment of the present invention, a higher-level Remote Procedure Call protocol generates transactions into messages that are sent over the mobile enhanced network 108 via the standard transport protocol(s). In this preferred embodiment, these mobile RPC messages contain the entire network transaction initiated by an application running on the Mobile End System 104, so it can be completed in its entirety by the Mobility Management Server. This enables the Mobility Management Server 102 and Mobile End System 104 to keep connection state information synchronized at all times—even during interruptions of network medium connectivity.

Each of Mobile End Systems 104 executes a mobility management software client that supplies the Mobile End System with the intelligence to intercept all network activity and relay it via the mobile RPC protocol to Mobility Management Server 102. In the preferred embodiment, the mobility management client works transparently with operating system features present on Mobile End Systems 104 (e.g., Windows NT, Windows 98, Windows 95, Windows CE, etc.) to keep client-site application sessions active when contact is lost with the network.

Mobility Management Server 102 maintains the state of each Mobile End System 104 and handles the complex session management required to maintain persistent connections to associated peer 108 such as host computer 110 attached to the other end of the connection end point. If a Mobile End System 104 becomes unreachable, suspends, or changes network address (e.g., due to roaming from one network interconnect to another), the Mobility Management Server 102 maintains the connection to the host system 110 or other connection end-point, by acknowledging receipt of

10

data and queuing requests. This proxy function means that the peer application never detects that the physical connection to the Mobile End System 104 has been lost—allowing the Mobile End System's application(s) to effectively maintain a continuous connection with its associated session end point (by simply and easily resuming operations once a physical connection again is established) despite the mobile system temporarily losing connection or roaming from one network interconnect 106A to another network interconnect 106K within coverage area 107K.

Mobility Management Server 102 also provides address management to solve the problem of Mobile End Systems 104 receiving different network addresses when they roam to different parts of the segmented network. Each Mobile End System 104 is provided with a virtual address on the primary network. Standard protocols or static assignment determine these virtual addresses. For each active Mobile End System 104, Mobility Management Server 102 maps the virtual address to the Mobile End System's current actual ("point of presence") address. While the point of presence address of a Mobile End System 104 may change when the device changes from one network segment to another, the virtual address stays constant while any connections are active or longer if the address is assigned statically.

Thus, the change of a point of presence address of a Mobile End System 104 remains entirely transparent to an associated session end point on host system 110 (or other peer) communicating with the Mobile End System via the Mobility Management Server 102. The peer 110 sees only the (unchanging) virtual address proxied by the server 102.

In the preferred embodiment, Mobility Management Server 102 can also provide centralized system management through console applications and exhaustive metrics. A system administrator can use these tools to configure and manage remote connections, and troubleshoot remote connection and system problems.

The proxy server function provided by Mobility Management Server 102 allows for different priority levels for network applications, users and machines. This is useful because each Mobility Management Server 102 is composed of finite processing resources. Allowing the system manager to configure the Mobility Management Server 102 in this way provides enhanced overall system and network performance. As one example, the system manager can configure Mobility Management Server 102 to allow real time applications such as streaming audio or video to have greater access to the Mobility Management Server 102's resources than other less demanding applications such as email.

In more detail, Mobility Management Server 102 can be configured via an application or application interface; standard network management protocols such as SNMP; a Web-based configuration interface; or a local user interface. It is possible to configure association priority and/or to configure application priority within an association. For example, the priority of each association relative to other associations running through the Mobility Management Server 102 is configurable by either the user name, or machine name (in the preferred embodiment, when the priority is configured for both the user and the machine that a user is logged in on, the configuration for the user may have higher precedence). In addition or alternatively, each association may have several levels of application priority, which is configured based on network application name. The system allows for any number of priority levels to exist. In one particular implementation, three priority levels are provided: low, medium and high.

US 6,981,047 B2

11

Server and Client Example Software Architecture

FIG 2 shows an example software architecture for Mobile End System 104 and Mobility Management Server 102. In accordance with one aspect of a presently preferred exemplary embodiment of the present invention, Mobile End System 104 and Mobility Management Server 102 run standard operating system and application software—with only a few new components being added to enable reliable and efficient persistent session connections over an intermittently connected mobile network 108. As shown in FIG 2, Mobile End System 104 runs conventional operating system software including network interface drivers 200, TCP/UDP transport support 202, a transport driver interface (TDI) 204, and a socket API 206 used to interface with one or more conventional network applications 208. Conventional network file and print services 210 may also be provided to communicate with conventional TDI 204. Mobility Management Server 102 may include similar conventional network interface drivers 200, TCP/UDP transport support 202, a transport driver interface (TDI) 204, and a socket API 206 used to interface with one or more conventional network applications 208. Mobile End System 104 and Mobility Management Server 102 may each further include conventional security software such as a network/Security provider 236 (Mobile End System) and a user/security database 238 (server).

In accordance with one exemplary aspect of the present invention, a new, mobile interceptor component 212 is inserted between the TCP/UDP transport module 202 and the transport driver interface (TDI) 204 of the Mobile End System 104 software architecture. Mobile interceptor 212 intercepts certain calls at the TDI 204 interface and routes them via RPC and Internet Mobility Protocols and the standard TCP/UDP transport protocols 202 to Mobility Management Server 102 over network 108. Mobile interceptor 212 thus can intercept all network activity and relay it to server 102. Interceptor 212 works transparently with operating system features to allow client-side application sessions to remain active when the Mobile End System 104 loses contact with network 108.

While mobile interceptor 212 could operate at a different level than the transport driver interface 204 (e.g., at the socket API level 206), there are advantages in having mobile interceptor 212 operate at the TDI level. Many conventional operating systems (e.g., Microsoft Windows 95, Windows 98, Windows NT and Windows CE) provide TDI interface 204—thus providing compatibility without any need to change operating system components. Furthermore, because the transport driver interface 204 is a kernel level interface, there is no need to switch to user mode—thus realizing performance improvements. Furthermore, mobile interceptor 212 working at the level of TDI interface 204 is able to intercept from a variety of different network applications 208 (e.g., multiple simultaneously running applications) as well as encompassing network file and print services 210 (which would have to be handled differently if the interceptor operated at the socket API level 206 for example).

FIG 2A shows an example high level flowchart of how mobile interceptor 212 works. A call to the TDI interface 204 of Mobile End System 104 (block 250) is intercepted by mobile interceptor 212 (block 252). Mobile interceptor 212 packages the intercepted RPC call in a fragment in accordance with an Internet Mobility Protocol, and sends the fragment as a datagram via a conventional transport protocol such as UDP or TCP over the LAN, WAN or other transport 108 to Mobility Management Server 102 (block 252). The Mobility Management Server 102 receives and unpackages

12

the RPC datagram (block 254), and provides the requested service (for example, acting as a proxy to the Mobile End System application 208 by passing data or a response to an application server process running on Fixed End System 110).

Referring once again to FIG. 2, Mobility Management Server 102 includes an address translator 220 that intercepts messages to/from Mobile End Systems 104 via a conventional network interface driver 222. For example, address translator 230 recognizes messages from an associated session peer (Fixed End System 110) destined for the Mobile End System 104 virtual address. These incoming Mobile End System messages are provided to proxy server 224, which then maps the virtual address and message to previously queued transactions and then forwards the responses back to the current point of presence addresses being used by the associated Mobile End System 104.

As also shown in FIG. 2, Mobility Management Server 102 includes, in addition to address translation (intermediate driver) 220, and proxy server 224, a configuration manager 228, a control/user interface 230 and a monitor 232. Configuration management 228 is used to provide configuration information and parameters to allow proxy server 224 to manage connections. Control, user interface 230 and monitor 232 allow a user to interact with proxy server 214. Mobile Interceptor

FIG 3 shows an example software architecture for mobile interceptor 212 that support the RPC Protocol and the Internet Mobility Protocol in accordance with a presently preferred exemplary embodiment of the present invention. In this example, mobile interceptor 212 has two functional components:

- a Remote Procedure Call protocol engine 240; and
- an Internet Mobility Protocol engine 244.

Mobile interceptor 212 in the preferred embodiment thus supports Remote Procedure Call protocol and Internet Mobility Protocol to connect Mobility Management Server 102 to each Mobile End System 104. Remote procedure calls provide a method for allowing a process on a local system to invoke a procedure on a remote system. Typically, the local system is not aware that the procedure call is being executed on a remote system. The use of RPC protocols allows Mobile End System 104 to go out of range or suspend operation without losing active network sessions. Since session maintenance does not depend on a customized application, off-the-shelf applications will run without modification in the mobile environment of network 108.

Network applications typically use application-level interfaces such as Windows sockets. A single call to an application-level API may generate several outgoing or incoming data packets at the transport, or media access layer. In prior mobile networks, if one of these packets is lost, the state of the entire connection may become ambiguous and the session must be dropped. In the preferred exemplary embodiment of the present invention providing RPCs, the Mobility Management Server 102 and the Mobile End Systems 104 share sufficient knowledge of the connection state to maintain a coherent logical link at all times—even during physical interruption.

The Internet Mobility Protocol provided in accordance with a presently preferred exemplary embodiment of the present invention compensates for differences between wire-line and other less reliable networks such as wireless. Adjusted frame sizes and protocol timing provide significant performance improvements over non-mobile-aware transports—dramatically reducing network traffic. This is important when bandwidth is limited or when battery life is a concern.

US 6,981,047 B2

13

The Internet Mobility Protocol provided in accordance with a presently preferred embodiment of the present invention also ensure the security of organization's data as it passes between the Mobile End System 104 and the Mobility Management Server 102 on the public wire-line networks or airway. The Internet Mobility Protocol provides a basic firewall function by allowing only authenticated devices access to the organizational network. The Internet Mobility Protocol can also certify and encrypt all communications between the mobility management system 102 and the Mobile End System 104.

The Remote Procedure Call protocol engine 240 on Mobile End System 104 of FIG 3 marshals TDI call parameters, formats the data, and sends the request to the Internet Mobility Protocol engine 244 for forwarding to Mobility Management Server 102 where the TDI Remote Procedure Call engine 240' executes the calls. Mobile End Systems 104 marshal TDI call parameters according to the Remote Procedure Call protocol. When the Mobility Management Server 102 TDI Remote Procedure Call protocol engine 240' receives these RPCs, it executes the calls on behalf of the Mobile End System 104. The Mobility Management Server 102 TDI Remote Procedure Call protocol engine 240' shares the complete network state for each connected Mobile End System with the peer Mobile End System 104's RPC engine 240. In addition to performing remote procedure calls on behalf of the Mobile End Systems 104, the server RPC engine 240' is also responsible for system flow control, remote procedure call parsing, virtual address multiplexing (in coordination with services provided by address translator 220), remote procedure call transaction prioritization, scheduling, and coalescing.

The Internet Mobility Protocol engine 244 performs reliable datagram services, sequencing, fragmentation, and re-assembly of messages. It can, when configured, also provide authentication, certification, data encryption and compression for enhanced privacy, security and throughput. Because the Internet Mobility Protocol engine 244 functions in power-sensitive environments using several different transports, it is power management aware and is transport independent.

FIG 3A shows an example process mobile interceptor 212 performs to communicate a TDI call to Mobility Management Server 102. Generally, the mobile interceptor RPC protocol engine 240 forwards marshaled TDI calls to the Internet Mobility Protocol engine 244 to be transmitted to the Mobility Management Server 102. RPC protocol engine 240 does this by posting the RPC call to a queue maintained by the Internet Mobility Protocol engine 244 (block 302). To facilitate bandwidth management, the Internet Mobility Protocol engine 244 delays sending received RPC calls for some period of time ("the RPC coalesce time out period") (block 304). Typically, the RPC coalesce timeout is set between five and fifteen milliseconds as one example but is user configurable. This delay allows the RPC engine 240 to continue posting TDI calls to the Internet Mobility Protocol engine 244 queue so that more than one RPC call can be transmitted to the Mobility Management Server 102 in the same datagram (fragment).

When the coalesce timer expires, or the RPC protocol engine 240 determines that it will not be receiving more RPC calls (decision block 306), the RPC engine provides the Internet Mobility Protocol engine 244 with a request to flush the queue, coalesce the RPC calls into a single frame, and forward the frame to its peer (block 308). This coalescing reduces the number of transmissions—enhancing protocol performance.

14

As mentioned above, Mobility Management Server 102 proxy server also has an RPC protocol engine 212' and an Internet Mobility Protocol engine 244'. FIG 3B shows an example process performed by Mobility Management Server 102 upon receipt of an Internet Mobility Protocol message frame from Mobile End System 104. Once the frame is received by the Mobility Management Server 102, the Internet Mobility Protocol engine 244' reconstructs the frame if fragmented (due to the maximum transmission size of the underlying transport) and then demultiplexes the contents of the message to determine which Mobile End System 104 it was received from. This demultiplexing allows the Internet Mobility Protocol engine 244' to provide the Remote Procedure Call engine 240' with the correct association-specific context information.

The Internet Mobility Protocol engine 244' then formulates the received message into a RPC receive indication system work request 354, and provides the Mobility Management Server 102 RPC engine 240' with the formulated work request and association-specific context information. When RPC protocol engine 240' receives work request 352, it places it into an association-specific work queue 356, and schedules the association to run by providing a scheduled request to a global queue 358. The main work thread of RPC engine 240' is then signaled that work is available. Once the main thread is awake, it polls the global queue 358 to find the previously queued association scheduled event. It then de-queues the event and begins to process the association-specific work queue 356.

On the association specific work queue 356 it finds the previously queued RPC receive indication work request. The main thread then de-queues the RPC receive indication work request 356 and parses the request. Because of the coalescing described in connection with FIG 3A, the Mobility Management Server 102 often receives several RPC transactions bundled in each datagram. It then demultiplexes each RPC transaction back into distinct remote procedure calls and executes the requested function on behalf of Mobile End System 104. For performance purposes RPC engine 240' may provide a look ahead mechanism during the parsing process of the RPC messages to see if it can execute some of the requested transactions concurrently (pipelining). How RPC Protocol Engine 240' Runs RPC Associations.

FIG 4 is a flowchart of an example process for running RPC associations placed on an association work queue 356. When an RPC association is scheduled to run, the main thread for the RPC protocol engine 240' (which may be implemented as a state machine) de-queues the work request from global work queue 358 and determines the type of work request.

There are six basic types of RPC work requests in the preferred embodiment:

- schedule request;
- connect indication;
- disconnect indication;
- local terminate association;
- "resources available" request; and
- ping inactivity timeout.

RPC protocol engine 240' handles these various types of requests differently depending upon their type. RPC protocol engine 240' tests the request type (indicated by information associated with the request as stored on global queue 358) in order to determine how to process the request.

If the type of work request is a "schedule request" (decision block 360), the RPC engine 240' determines which association is being scheduled (block 362). RPC engine 240'

US 6,981,047 B2

15

can determine this information from what is stored on global queue 358. Once the association is known, RPC engine 240' can identify the particular one of association work queues 356(1) . . . 356(n) the corresponding request is stored on RPC engine 362 retrieves the corresponding association control block (block 362), and calls a Process Association Work task 364 to begin processing the work in a specific association's work queue 356 as previously noted

FIG 5 shows example steps performed by the "process association work" task 364 of FIG. 4. Once the specific association has been determined, this "process association work" task 364 is called to process the work that resides in the corresponding association work queue 356. If the de-queued work request (block 390) is an RPC receive request (decision block 392), it is sent to the RPC parser to be processed (block 394). Otherwise, if the de-queued work request is a pending receive request (decision block 396), the RPC engine 240' requests TDI 204' to start receiving data on behalf of the application's connection (block 398). If the de-queued work request is a pending connect request (decision block 400), RPC engine 240' requests TDI 204' to issue an application specified TCP (or other transport protocol) connect request (block 402). It then waits for a response from the TDI layer 204'. Once the request is completed by TDI 204', its status is determined and then reported back to the original requesting entity. As a performance measure, RPC engine 240' may decide to retry the connect request process some number of times by placing the request back on the associations-specific work queue (356) before actually reporting an error back to the requesting peer. This again is done in an effort to reduce network bandwidth and processing consumption.

The above process continues to loop until a "scheduling weight complete" test (block 404) is satisfied. In this example, a scheduling weight is used to decide how many work requests will be de-queued and processed for this particular association. This scheduling weight is a configuration parameter set by configuration manager 228, and is acquired when the association connect indication occurs (FIG 4, block 372). This value is configurable based on user or the physical identification of the machine.

Once the RPC engine is finished with the association work queue 356 (for the time at least), it may proceed to process dispatch queues (block 406) (to be discussed in more detail below). If, after processing work on the association's work queue 356, more work remains in the association work queue, the RPC engine 240' will reschedule the association to run again at a later time by posting a new schedule request to the global work queue 358 (FIG. 4, decision block 366, block 368; FIG 5, decision block 408, block 410).

Referring once again to FIG. 4, if the RPC work requested is a "connect indication" (decision block 370), RPC engine 240' is being requested to instantiate a new association with a mobile peer (usually, but not always, the Mobile End System 104). As one example, the connect indication may provide the RPC engine 240' with the following information about the peer machine which is initiating the connection:

- physical identifier of the machine,
- name of the user logged into the machine,
- address of the peer machine, and
- optional connection data from the peer RPC engine 240.

In response to the connect indication (decision block 370), the RPC engine 240 calls the configuration manager 228 with these parameters. Configuration manager 228 uses these parameters to determine the exact configuration for the new connection. The configuration (e.g., association sched-

16

uling weight and the list of all applications that require non-default scheduling priorities along with those priorities) is then returned to the RPC engine 240' for storage and execution. RPC engine 240' then starts the new association, and creates a new association control block (block 372). As shown in FIG. 5A the following actions may be taken:

- allocate and association control block (block 372A);
- initialize system wide resources with defaults (block 372B);
- get configuration overrides with current configuration settings (block 372C);
- initialize flags (block 372D);
- initialize the association-specific work queue (block 372E);
- initialize association object hash table (block 372F);
- initialize the coalesce timer (block 372G); and
- insert association control block into session table (block 372H).

A "disconnect indication" is issued by the Internet Mobility Protocol engine 244' to the RPC engine 240' when the Internet Mobility Protocol engine has determined that the association must be terminated. The RPC engine 240' tests for this disconnect indication (block 374), and in response, stops the association and destroys the association control block (block 376). As shown in FIG. 5B, the following steps may be performed:

- mark the association as deleted to prevent any further processing of work that may be outstanding (block 376A);
- close all associated association objects including process, connection and address objects (block 376B);
- free all elements on work queue (block 376C);
- stop coalesce timer if running (block 376D);
- decrement association control block reference count (block 376E); and
- if the reference count is zero (tested for by block 376F):
 - =destroy association specific work queue,
 - =destroy object hash table,
 - =destroy coalesce timer,
 - =remove association control block from association table, and
 - =free control block (376G).

A "terminate session" request is issued when system 100 has determined that the association must be terminated. This request is issued by the system administrator, the operating system or an application. RPC engine 240' handles a terminate session request in the same way it handles a disconnect request (decision block 378, block 376).

In the preferred embodiment, the interface between the RPC engine 240' and the Internet Mobility Protocol engine 244' specifies a flow control mechanism based on credits. Each time one thread posts a work request to another thread, the call thread returns the number of credits left in the work queue. When a queue becomes full, the credit count goes to zero. By convention, the calling thread is to stop posting further work once the credit count goes to zero. Therefore, it is necessary to have a mechanism to tell the calling thread that "resources are available" once the queued work is processed and more room is available by some user configurable/pre-determined low-water mark in the queue. This is the purpose of the "resources available" work indication (tested for by decision block 380). As shown in FIG. 5C, the following steps may be performed when the credit count goes to zero:

US 6,981,047 B2

17

mark association as "low mark pending" by setting the RPC_LMPQ_SEND_FLAG (block 379A) Once in this state:

all received datagrams are discarded (block 379B);

all received stream events are throttled by refusing to accept the data (block 379C) (this causes the TCP or other transport receive window to eventually close, and provides flow control between the Fixed End System 110 and the Mobility Management Server 102; before returning, the preferred embodiment jams a "pending receive request" to the front of the association specific work queue 356 so that outstanding stream receive event processing will continue immediately once resources are made available)

all received connect events are refused for passive connections (block 379D)

When the "resources available" indication is received by the RPC engine 240' (FIG 4, decision block 380), the RPC engine determine whether the association has work pending in its associated association work queue 356; if it does, the RPC engine marks the queue as eligible to run by posting the association to the global work queue 358 (block 382) If a pending receive request has been posted during the time the association was in the low mark pending state, it is processed at this time (in the preferred embodiment, the RPC engine 240' continues to accept any received connect requests during this processing).

Referring once again to FIG 4, if RPC engine 240' determines that the Mobility Management Server 102 channel used for "ping" has been inactive for a specified period of time (decision block 384), the channel is closed and the resources are freed back to the system to be used by other processes (block 386)

RPC Parsing and Priority Queuing

Referring back to FIG 5, it was noted above that RPC engine parsed an RPC receive request upon receipt (see blocks 392, block 394) Parsing is necessary in the preferred embodiment because a single received datagram can contain multiple RPC calls, and because RPC calls can span multiple Internet Mobility Protocol datagram fragments. An example format for an RPC receive work request 500 is shown in FIG 6 Each RPC receive work request has at least a main fragment 502(1), and may have any number of additional fragments 502(2) - 502(N). Main fragment 502(1) contains the work request structure header 503 and a receive overlay 504 The receive overlay 504 is a structure overlay placed on top of the fragment 502(1) by the Internet Mobility Protocol engine 244 Within this overlay 504 is a structure member called pUserData that points to the first RPC call 506(1) within the work request 500

The FIG 6 example illustrates a work request 500 that contains several RPC calls 506(1), 506(2) - 506(8) As shown in the FIG 6 example, an RPC work request 500 need not be contained in a contiguous block of memory or in a single fragment 502 In the example shown, a second fragment 502(2) and a third fragment 502(3) that are chained together to the main fragment 502(1) in a linked list

Thus, RPC parser 394 in this example handles the following boundary conditions:

each RPC receive request 500 may contain one or more RPC calls;

one or more RPC calls 506 may exist in a single fragment 502;

each RPC call 506 may exist completely contained in a fragment 502; and

each RPC call 506 may span more than one fragment 502

18

FIG. 7 shows an example RPC parser process 394 to parse an RPC receive work request 500. In this example, the RPC parser 394 gets the first fragment 502(1) in the work request, gets the first RPC call 506(1) in the fragment, and parses that RPC call Parser 394 proceeds through the RPC receive work request 500 and processes each RPC call 506 in turn. If the number of fragment bytes remaining in the RPC receive work request 500 fragment 502(1) is greater than the size of the RPC header 503, parser 394 determines whether the RPC call is fully contained within the RPC fragment 502 and thus may be processed (this may be determined by testing whether the RPC call length is greater than the number of fragment bytes remaining) If the RPC call type is a chain exception, then the RPC call will handle the updating of the RPC parser 394 state. In the proxy server 224, the only RPC calls using the chain exception are the "datagram send" and "stream send" calls. This chain exception procedure is done to allow the RPC engine to avoid fragment copies by chaining memory descriptor lists together for the purpose of RPC send calls

Once the parser 394 identifies an RPC call type, a pointer to the beginning of the RPC information is passed to the RPC engine 240 for execution. The RPC engine divides all TDI procedure calls into different priorities for execution. The highest priority calls are immediately executed by passing them to an RPC dispatcher 395 for immediate execution. All lower priority calls are dispatched to dispatch queues 510 for future processing. Each dispatch queue 510 represents a discrete priority

In the preferred embodiment, mobile applications call the "open address" object and "open connection" object functions before executing other TDI networking functions. Therefore, the system assigns application level priorities during the "open address" object and "open connection" object calls. In the example embodiment, once an address or connection object is assigned a priority, all calls that are associated with that object are executed within that assigned priority

If, for example, the RPC call is a TDI Open Address Object request or a TDI Open Connection Object Request, it is sent to the RPC dispatcher 395 for immediate execution. The Open Address and Open Connection object RPC calls provide access to a process ID or process name that are used to match against the information provided by the configuration manager 228 during the configuration requests that occurs within the association connect indication described earlier. This is used to acquire configuration for the address or connection object

In the preferred embodiment, all RPC calls have at least an address object or connection object as a parameter. When the call is made, the priority assigned to that specific object is used as the priority for the RPC call. The configuration assigned to the address or connection object determines which priority all associated RPC calls will be executed in. For example, if the assigned priority is "high," all RPC calls will be executed immediately without being dispatched to a dispatch queue 510. If the assigned priority is "1," all RPC calls will be placed into dispatch queue 510(1)

Referring once again to FIG 5, once the "process association work" task 364 process has completed executing its scheduled amount of association work (decision block 404), it checks to see if the dispatch queues require servicing (block 406) FIG 8 is a flowchart of example steps performed by the "process dispatch queues" block 406 of FIG 5 to process the dispatch queues 510 shown in FIG 7

In this example, dispatch queues 510 are processed beginning with the highest priority queue (510(1) in this example)

US 6,981,047 B2

19

(block 408) Each queue 510 is assigned a weight factor. The weight factor is a configuration parameter that is returned by the configuration manager 228 when a Mobile End System 104 to Mobility Management Server 102 association is created. As one example, low priority dispatch queues 510 can have a weight factor of 4, and medium priority queues can have a weight factor of 8. High priority RPC calls do not, in this example, use weight factors because they are executed immediately as they are parsed.

RPC engine 240' loops through the de-queuing of RPC calls from the current queue until either the queue is empty or the queue weight number of RPC calls has been processed (blocks 412-416). For each de-queued RPC call, the RPC dispatcher 395 is called to execute the call. The RPC dispatcher 395 executes the procedural call on behalf of the Mobile End System 104, and formulates the Mobile End System response for those RPC calls that require responses.

If, after exiting the loop, the queue still has work remaining (decision block 418), the queue will be marked as eligible to run again (block 420). By exiting the loop, the system yields the processor to the next lower priority queue (blocks 424, 410). This ensures that all priority levels are given an opportunity to run no matter how much work exists in any particular queue. The system gets the next queue to service, and iterates the process until all queues have been processed. At the end of processing all queues, the system tests to see if any queues have been marked as eligible to run—and if so, the association is scheduled to run again by posting a schedule request to the global work queue. The association is scheduled to run again in the "process global work" routine shown in FIG 4 above. This approach yields the processor to allow other associations that have work to process an opportunity run. By assigning each queue a weight factor, the system may be tuned to allow different priority levels unequal access to the Mobility Management Server 102's CPU. Thus, higher priority queues are not only executed first, but may also be tuned to allow greater access to the CPU.

Mobility Management Server RPC Responses

The discussion above relates explains how remote procedure calls are sent from the Mobile End System 104 to the Mobility Management Server 102 for execution. In addition to this type of RPC call, the Mobility Management Server 102 RPC engine 240' also supports RPC events and RPC receive responses. These are RPC messages that are generated asynchronously as a result of association specific connection peer activity (usually the Fixed End System 110). Mobility Management Server 102 RPC engine 240' completes RPC transactions that are executed by the RPC dispatcher 395. Not all RPC calls require a response on successful completion. Those RPC calls that do require responses on successful completion cause the RPC dispatcher 395 to build the appropriate response and post the response to the Internet Mobile Protocol engine 244' to be returned to the peer Mobile End System 104. All RPC calls generate a response when the RPC call fails (the RPC receive response is the exception to above).

RPC events originate as a result of network 108 activity by the association specific connection (usually the [j1]Fixed End System 110). These RPC event messages are, in the preferred embodiment, proxied by the Mobility Management Server 102 and forwarded to the Mobile End System 104. The preferred embodiment Mobility Management Server 102 supports the following RPC event calls:

Disconnect Event (this occurs when association-specific connected peer (usually the Fixed End System 110) issues a transport level disconnect request; the discon-

20

nect is received by the proxy server 224 on behalf of the Mobile End System 104, and the proxy server then transmits a disconnect event to the Mobile End System);

Stream Receive Event (this event occurs when the association-specific connected peer (usually the Fixed End System 110) has sent stream data to the Mobile End System 104; the proxy server 224 receives this data on behalf of the Mobile End System 104, and sends the data to the Mobile End System in the form of a Receive Response);

Receive Datagram Event (this event occurs when any association-specific portal receives datagrams from a network peer (usually the Fixed End System 110) destined for the Mobile End System 104 through the Mobility Management Server 102; the proxy server 224 accepts these datagrams on behalf of the Mobile End System, and forwards them to the Mobile End System in the form of receive datagram events; and

Connect Event (this event occurs when the association-specific listening portal receives a transport layer connect request (usually from the Fixed End System 110) when it wishes to establish a transport layer end-to-end connection with a Mobile End System 104; the proxy server 224 accepts the connect request on behalf of the Mobile End System, and then builds a connect event RPC call and forwards it to the Mobile End System).

FIG 9 shows how the RPC engine 240' handles proxy server-generated RPC calls. For high priority address and connection objects, the RPC engine 240' dispatches a send request to the Internet Mobility Protocol engine 244' immediately. The send request results in forwarding the RPC message to the peer Mobile End System 104. For lower priority objects, the Internet Mobility Protocol engine 244' send request is posted to an appropriate priority queue 510'. If the association is not scheduled to run, a schedule request is also posted to the global queue 358'. The Internet Mobility Protocol send request is finally executed when the dispatch queues are processed as described earlier in connection with FIGS 5 & 8.

Internet Mobility Protocol

Internet Mobility Protocol provided in accordance with an example embodiment of the present invention is a message oriented connection based protocol. It provides guaranteed delivery, (re)order detection, and loss recovery. Further, unlike other conventional connection oriented protocols (i.e. TCP), it allows for multiple distinct streams of data to be combined over a single channel; and allows for guaranteed, unreliable, as well as new message oriented reliable data to traverse the network through the single virtual channel simultaneously. This new message oriented level of service can alert the requester when the Internet Mobility Protocol peer has acknowledged a given program data unit.

The Internet Mobility Protocol provided in accordance with a presently preferred exemplary embodiment of the present invention is designed to be an overlay on existing network topologies and technologies. Due to its indifference to the underlying network architecture, it is transport agnostic. As long as there is a way for packetized data to traverse between two peers, Internet Mobility Protocol can be deployed. Each node's network point of presence (POP) or network infrastructure can also be changed without affecting the flow of data except where physical boundary, policy or limitations of bandwidth apply.

With the help of the layer above, Internet Mobility Protocol coalesces data from many sources and shuttles the data between the peers using underlying datagram facilities. As

US 6,981,047 B2

21

each discrete unit of data is presented from the upper layer, Internet Mobility Protocol combines into a single stream and subsequently submits it for transmission. The data units are then forwarded to the peer over the existing network where upon reception, with the help from the layer above, the stream is demultiplexed back into multiple distinct data units. This allows for optimum use of available bandwidth, by generating the maximum sized network frames possible for each new transmission. This also has the added benefit of training the channel once for maximum bandwidth utilization and have its parameters applied to all session level connections.

In rare instances where one channel is insufficient, the Internet Mobility Protocol further allows multiple channels to be established between the peers—thus allowing for data prioritization and possibly providing a guaranteed quality of service (if the underlying network provides the service).

The Internet Mobility Protocol also provides a dynamically selectable guaranteed or unreliable levels of service. For example, each protocol data unit that is submitted for transmission can be queued with either a validity time period or a number of retransmit attempts or both. Internet Mobility Protocol will expire a data unit when either threshold is reached, and remove it from subsequent transmission attempts.

Internet Mobility Protocol's additional protocol overhead is kept minimal by use of a variable length header. The frame type and any optional fields determine the size of the header. These optional fields are added in a specific order to enable easy parsing by the receiving side and bits in the header flag field denote their presence. All other control and configuration information necessary for the peers to communicate can be passed through the in-band control channel. Any control information that needs to be sent is added to the frame prior to any application level protocol data unit. The receiving side processes the control information and then passes the rest of the payload to the upper layer.

Designed to run over relatively unreliable network links where the error probability is relatively high, Internet Mobility Protocol utilizes a number of techniques to insure data integrity and obtain optimum network performance. To insure data integrity, a Fletcher checksum algorithm is used to detect errant frames. This algorithm was selected due to the fact of its efficiency as well as its detection capability. It can determine not only bit errors, but also bit reordering.

Sequence numbers are used to insure ordered delivery of data. Internet Mobility Protocol sequence numbers do not, however, represent each byte of data as in TCP. They represent a frame of data that can be, in one example implementation, as large as 65535 bytes (including the Internet Mobility Protocol header). They are 32 bits or other convenient length in one example to insure that wrap-around does not occur over high bandwidth links in a limited amount of time.

Combining this capability along with the expiration of data, retransmitted (retried) frames may contain less information than the previous version that was generated by the transmitting side. A frame id is provided to enable detection of the latest versioned frame. However, since data is never added in the preferred embodiment and each element removed is an entire protocol data unit, this is not a necessity. In one example, the Internet Mobility Protocol will only process the first instance of a specific frame it receives—no matter how many other versions of that frame are transmitted. Each frame created that carries new user payload is assigned its own unique sequence number.

Performance is gained by using of a sliding window technique—thus allowing for more than one frame to be

22

outstanding (transmitted) at a time before requiring the peer to acknowledge reception of the data. To insure timely delivery of the data, a positive acknowledgement and timer based retransmit scheme is used. To further optimize the use of the channel, a selective acknowledgement mechanism is employed that allows for fast retransmission of missing frames and quick recovery during lossy or congested periods of network connectivity. In one example, this selective acknowledgement mechanism is represented by an optional bit field that is included in the header.

A congestion avoidance algorithm is also included to allow the protocol to back off from rapid retransmission of frames. For example, a round trip time can be calculated for each frame that has successfully transfer between the peers without a retransmit. This time value is averaged and then used as the basis for the retransmission timeout value. As each frame is sent, a timeout is established for that frame. If an acknowledgement for that frame is not received, and the frame has actually been transmitted, the frame is resent. The timeout value is then increased and then used as the basis for the next retransmission time. This retransmit time-out is bounded on both the upper and lower side to insure that the value is within a reasonable range.

Internet Mobility Protocol also considers the send and receive paths separately. This is especially useful on channels that are asymmetric in nature. Based on hysteresis, the Internet Mobility Protocol automatically adjusts parameters such as frame size (fragmentation threshold), number of frames outstanding, retransmit time, and delayed acknowledgement time to reduce the amount of duplicate data sent through the network.

Due to the fact that Internet Mobility Protocol allows a node to migrate to different points of attachment on diverse networks, characteristics (e.g., frame size) of the underlying network may change midstream. An artifact of this migration is that frames that have been queued for transmission on one network may no longer fit over the new medium the mobile device is currently attached to. Combining this issue with the fact that fragmentation may not be supported by all network infrastructures, fragmentation is dealt with at the Internet Mobility Protocol level. Before each frame is submitted for transmission, Internet Mobility Protocol assesses whether or not it exceeds the current fragmentation threshold. Note that this value may be less than the current maximum transmission unit for performance reason (smaller frames have a greater likelihood of reaching its ultimate destination than larger frames). The tradeoff between greater protocol overhead versus more retransmissions is weighed by Internet Mobility Protocol, and the frame size may be reduced in an attempt to reduce overall retransmissions. If a given frame will fit, it is sent in its entirety. If not, the frame is split into maximum allowable size for the given connection. If the frame is retransmitted, it is reassessed, and will be refragmented if the maximum transmission unit has been reduced (or alternatively, if the maximum transmission unit actually grew, the frame may be resent as a single frame without fragmentation).

The protocol itself is orthogonal in its design as either side may establish or terminate a connection to its peer. In a particular implementation, however, there may be a few minor operational differences in the protocol engine depending on where it is running. For example, based on where the protocol engine is running, certain inactivity detection and connection lifetime timeouts may be only invoked on one side. To allow administrative control, Internet Mobility Protocol engine running on the Mobility Management Server 102 keeps track of inactivity periods. If the specified